Honours Report: Tailoring Feedback in ITSs

Danny Rajendra

Abstract

The most effective way for students to learn is face-to-face with a qualified tutor. Intelligent Tutoring Systems (ITS) aim to provide a learning experience that approaches the standard of learning with a human tutor. This project looks at the factors why and how human tutors are so effective and how these factors can be implemented in SQL-Tutor. An evaluation of the changes was conducted. The evaluation showed that the changes made no significant contribution to the student's learning.

Contents

Chapter 1

Introduction

Computer Aided Instruction (CAI) systems are the first systems to attempt to tutor students using computers. The conventional CAI approach is to first determine how a good teacher would respond to each possible student action and then build a branching program with each response explicitly programmed. CAI systems typically deliver their instructions in a rigid and structured manner with no regard for the differences in the way individuals assimilate information [1]. Since CAI systems lack understanding of how individuals assimilate information, they frequently mismatch the system's internal processes and the student's cognitive processes. In order for a computer based educational system to achieve the same kind of individualized attention that a student would receive from a human tutor, it must be able to reason about the domain and the learner. This has prompted research in the field of Intelligent Tutoring Systems (ITSs).

They began as an attempt to address the deficiencies of CAI systems by adapting to the knowledge and needs of individual students. Intelligent Tutoring Systems (ITSs) are computer based training systems that incorporate techniques for communicating and transferring knowledge and skills to students. They provides a means to capture expert knowledge which the system uses to compose dynamic instructional interactions. The expert can program the knowledge used to make their decisions, rather than simply program the decision itself, devoid of content.

The major advantage that ITSs offer is individualized instruction without the expense of one-to-one tutoring. However, most ITSs are not yet able to do this as well as a human tutor. The objective of this project is to extend the capabilities of SQL-Tutor, an ITS used to tutor students in Structured Query Language(SQL), by trying to individualize feedback to match the student's level of ability.

1.1 Report Structure

Chapter 2 presents an introduction to the background work, describing ITSs related to this project, and SQL.

Chapter 3 identifies the most effective way for students to learn and examines the difficulties of trying to identify how human tutors individualize feedback to students.

In Chapter 4, four experienced tutors were interviewed. This section expands on the interview and the inferences that can be drawn from it. Based on the inferences drawn, appropriate changes are implemented.

Chapter 5 evaluates the changes to SQL-Tutor and a discussion of the results follow.

Finally, Section 6 presents the conclusion, and describes some ideas for further work.

Chapter 2

Background

There are two areas of research associated with this study: ITSs, in particular SQL-Tutor, and SQL.

2.1 Intelligent Tutoring Systems

ITSs originated from the artificial intelligence (AI) movement of the late 1950s and early 1960s. Then, workers such as Alan Turing, Marvin Minsky and John McCarthy thought that computers that could "think" as humans were just around the corner, and it seemed reasonable to assume that once "thinking" machines were created, they could perform any task associated with human thought, such as tutoring.

Researchers developed a number of CAI systems designed to present the student with a problem, and receive and record the student's response. However, these systems did not explicitly address the issues of how people learn and did not enjoy much success. By the early 1970s, many researchers shifted the focus from knowledge transfer to knowledge communication and this led to the birth of ITS. By making use of research in the field of Artificial Intelligence, ITSs were able to employ knowledge representation strategies to model a student's cognitive processes. Using an accurate model of both the student's and expert's knowledge, ITSs are able to provide instruction at the appropriate pace and level of abstraction for the student.

Although there is no standard architecture for ITSs, four components emerge from literature as typical of an ITS [2]. For the purpose of conceptualization and design, it is often easier to think of an ITS as consisting of several interdependent modules. A typical architecture [3] consists of: the domain module, the student model, the tutor module and the user interface module, as shown in Figure 2.1. The domain module contains domain knowledge needed for the generation of problems and /or solving the problem given to the student. The student model stores a model or road-map of the student's knowledge and is the most important part of an ITS, since,

in the teaching process, the student has to play a central role. The tutor module determines the way in which the ITS reacts, and, finally, the user interface module allows communication between the student and the ITS. Together, these modules generate and control the interaction between the ITS and the student.

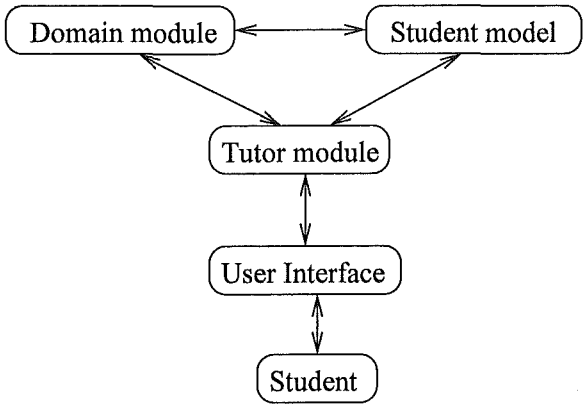

Figure 2.1: Typical architecture of ITSs

The range of programs that qualify as Intelligent Tutoring Systems is vast and can vary from primitive to sophisticated. In Section 2.3, a discussion of several ITSs is presented.

2.2 SQL and SQL-Tutor

Structured Query Language (SQL) is a language for accessing data in a relational database and was adopted as the industry standard in 1986 [4]. It is a simple, highly structured language. However, students are known to experience difficulties in learning it. The main problems with learning SQL by working with a database management system (DBMS) are that the error messages are limited to the syntax only, and that the DBMS is unable to deal with semantic errors [5]. With this in mind, SQL-Tutor has been developed for tutoring upper-level undergraduate students in SQL [6]. It is designed as a practise environment, where student are able to practise solving problems.

SQL-Tutor has a simple architecture, as shown in Figure 2.2. A screen-shot of the interface is illustrated in Figure 2.3. Domain knowledge is represented as constraints. Student modelling is undertaken by the constraint based modelling (CMB) module. The pedagogical module observes every action of the student. The solution entered by the student is sent to the CBM module which determines the relevant constraints. The violated constraints are updated to the student model (which is basically a list of violated

constraints). The pedagogical module then generates appropriate feedback. There are currently six levels of feedback implemented in SQL-Tutor, each determining how much information is provided to the student. The levels of feedback, arranged in increasing order of information content, are: positive or negative feedback, error flag, hint, partial solution, complete solution and all errors. A positive or negative feedback tells the students whether the solution is correct or not. An error flag informs the students about the clause in which the error occurred. A partial solution displays the correct solution for the incorrect clause. A complete solution displays the correct solution to the problem. Finally, an all errors feedback message lists the violated constraints for the problem. Figure 2.4 displays how the feedback level is selected in SQL-Tutor. When the student attempts a problem and does not provide a correct solution, he or she would receive a positive or negative feedback. If the student again attempts the same problem and gets it wrong, he or she would receive an error flag message. Finally, if the student again gets it wrong, a hint will be presented to the student. The other feedback levels (partial solution, complete solution and list all errors) are available only if the student chooses it.

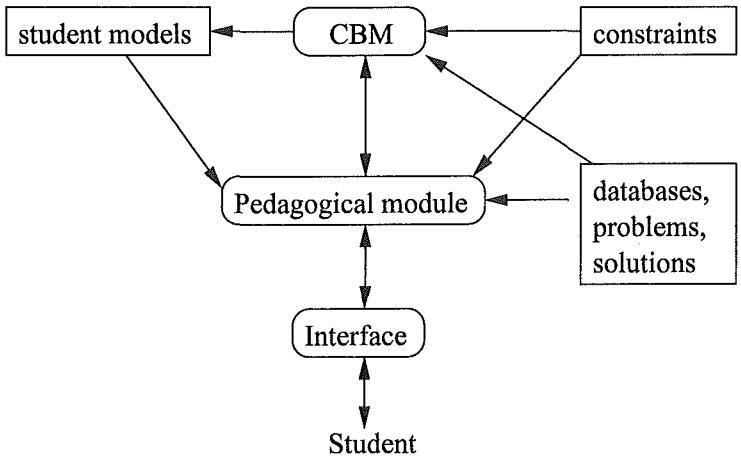

Figure 2.2: Architecture of SQL-Tutor

In SQL-Tutor, instruction can be individualized in several ways, by generating feedback dynamically, selecting topics and problems, and fading the scaffolding on the basis of the student model. This project focuses on generating feedback dynamically, in particular, trying to dynamically select feedback for each student based on his or her level of ability. The benefits of selecting feedback that matches the student's level of ability is that students learn better and faster.

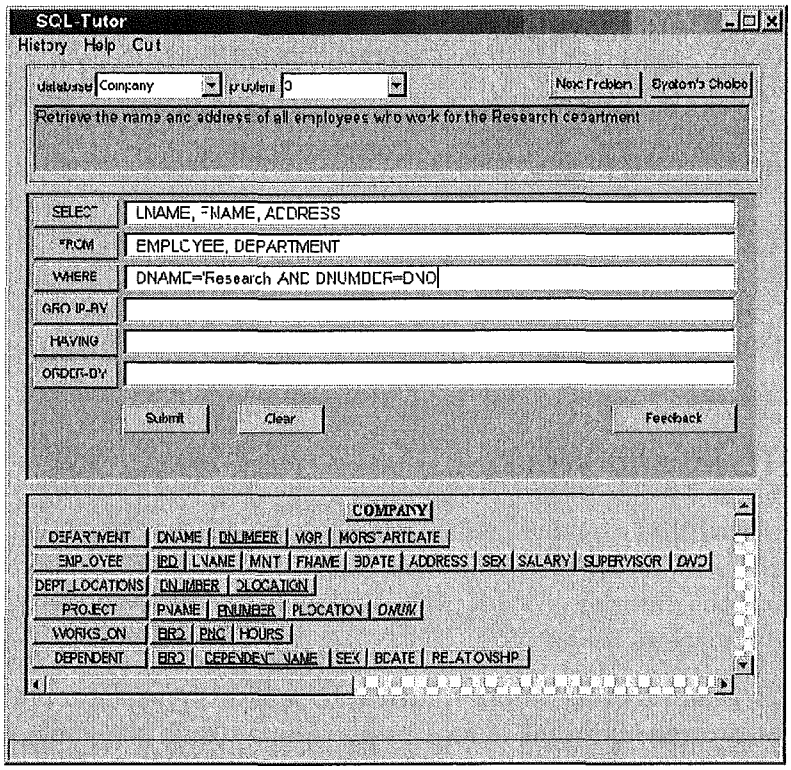

Figure 2.3: The interface of SQL-Tutor

All Students	
First Try	Positive/Negative
Second Try	Error Flag
Third Try	Hint
Avaliable only if requested	Partial Solution Complete Solution All Errors

Figure 2.4: Current feedback selection method in SQL-Tutor

2.3 Relevant Work

Much research has examined issues related to modelling the student's cognitive processes, rather than improving (individualizing) the quality of feedback from the tutoring system. Those that have improved feedback are briefly summarised below:

ANDES is an ITS that tutors students in classical Newtonian physics. It has two modules that it uses to provide feedback and hints tailored to the student's knowledge and goals [7]. The *helper* module tries to understand which plan or goals the student is pursuing as the student does an activity, and offers help specific to what it perceives to be helpful to the student's current plan. This means that ANDES individualizes feedback with respect to the individual's plan. If it detects an important misconception or a bad learning habit, it may engage the student in extensive multimedia instructional activities. The *assessor* module uses Bayesian reasoning to maintain a long-term model of the student's level of mastery of concepts and the student's preferred methods of problem solving and learning. The integration of the two modules means that ANDES can adapt its help as the student's level of knowledge changes over time.

RAPITS is an ITS that teaches introductory Pascal loop construction to

students from diverse backgrounds [8]. The system is aimed at novices in Pascal programming who might have a range of computing and cultural backgrounds. The aim of this system is to build a tutoring system with versatile teaching strategies. RAPITS uses a meta-teaching strategy where it determines the appropriate teaching strategy based on the student's history.

CIRCSIM-TUTOR is a natural language-based intelligent tutoring system to teach first-year medical students about the reflex control of blood pressure [9]. Students solve small problems and are tutored by Socratic dialogue with the computer. CIRCSIM-TUTOR individualizes feedback by searching for clue words in the student's response. For example, if the student does not understand the question and types "I do not *think* I understand the question", the system will look up the list of clue words and use the response associated with the particular clue word. CIRCSIM-TUTOR individualizes feedback with respect to the student's response.

MFD is an ITS that teaches arithmetic to fifth and sixth graders [10]. MFD learns the time an individual student requires to solve a problem by training a neural network, and then uses the neural network to make necessary predictions. Using the predictions, MDF will determine whether the student is not making sufficient progress on the current problem, and if so generates an easier problem. Besides intelligently selecting a topic on which a student should work on, and dynamically constructing problems that are appropriate for the student, MFD also provides hints that match the student's level of ability.

The common link between the four ITSs described above is that all adapt their feedback based on certain aspects of the student's behaviour. Despite the years in research and development in the field of ITSs, there are not many usable tutoring systems in use. This should encourage re-examination of the objectives and approaches to developing ITSs.

Chapter 3

The Human Tutor

Empirical studies have demonstrated that the most effective way for students to learn is to work alone, face-to-face with a qualified tutor, equipped with instructional material and laboratory equipment.

The main reason why human tutors are so effective is because they are able to tailor feedback (respond to student's questions about the subject matter, determine when students need help and identify the type of help needed) to students. Unfortunately, the need for qualified tutors makes this preferred form of instruction very costly and rare. ITSs aim to provide a learning experience for each student that approaches the standard of learning that he or she would receive in one-to-one tutoring from an expert tutor equipped with all the necessary training aids. The ideal ITS would be one modelled on a human tutor. However, trying to identify how human tutors individualize feedback to students is a complex topic because the process depends on many different factors that the tutor takes into account on any one occasion. For example, the tutor has several different ways of responding, and chooses an appropriate one depending on:

- What model of the learning process the tutor has in mind for each outcome of the instruction, and what steps the tutor thinks the student needs to follow to reach the outcome.
- Where the tutor thinks the student is in the sequence of steps the student needs to follow.
- The tutor's diagnosis of why the student might have made a mistake (which is based on the model of the student's abilities, background knowledge, and personality, that the tutor has in mind).

All this varies with different kinds of subject matter, how easy or difficult the tutor thinks the subject matter is, and how much time and effort the tutor thinks the subject matter deserves. It also varies with the tutor's estimate of the motivational level of the student.

The main goal of ITSs is to provide a learning experience for each student that approaches the standard of learning that he or she would receive in one-to-one tutoring from an expert tutor. One method of achieving this is to provide indivualization of instructions (providing instructions that match the student's level of ability). Individualization of instructions is important because students may have different prior knowledge, learning experience, particular interests, motivation, or personality, and these factors affect how a human tutor interacts with the student. It is proposed that SQL-Tutor be modified so that it would better individualize feedback to different students. In order to gain a better understanding of how human tutors individualize instructions or feedback, and what factors they take into account during the process of adapting instructions or feedback, several tutors were interviewed.

Chapter 4

The Interviews

Four tutors, each with several years of tutoring in database topics were interviewed. Only four tutors were interviewed because they had both the tutoring experience and domain expertise. Unskilled tutors may have domain expertise, but lack the tutoring experience that is often crucial to the successful interactions between the tutor and the student. The tutors were given a brief description of the project and its aims, before being interviewed. Each interview session lasted for approximately fifteen minutes, and a total of thirteen questions were asked. The questions centred around the interactions between the tutor and the student; in particular how the tutor decides what instruction or feedback to provide to the student (Appendix A). An example of a question asked is : What factors do you take into account when deciding how to reply? Section 4.1 reports the findings of the interviews and how it can apply to SQL-Tutor.

4.1 Possible Improvements

After analysing the answers given by the tutors, there emerged several possible improvements that can be made to SQL-Tutor.

Using examples

Firstly, tutors usually explain by giving examples and explaining each step of the example. By providing the student with an example that is related to the question the student is trying to solve, it allows the students to analogically infer an explanation for the new situation. With the tutor explaining each step of the example, students are also able to view the (analogous) solution path. SQL-Tutor does not have such a feedback and this provides motivation for this feedback method to be introduced. The best way to illustrate how this works is with an example. Figure 4.1 illustrates a situation in which problem 1 requires the student to specify a query.

Problem 1:
Retrieve the last name and address of all employees who work for the Research department.

Correct solution :	Student solution:
Select lname, address	Select lname
From employee, department	From employee, department
Where dname='Research' and dnumber=dno	Where dname='Research'

Feedback messages provided by SQL-Tutor

Positive/Negative Feedback : Good try – but there are some mistakes
Error Flag : Almost there – a few mistakes though. One of them is in the where clause
Hint : Almost there – you made 2 mistakes. If there are N tables in the from clause, then there should be N–1 join conditions in the where clause.
Partial Solution : Almost there – you made 2 mistakes. One of them is in the where clause. It should be : where dname='Research' and dnumber=dno
List all errors : 1. If there are N tables in the from clause. then there should be N–1 join conditions in where clause.
2. Check the expressions in the select clause!
Complete solution : The correct solution of this problem is: select lname, address
from employee, department
where dname='Research' and dnumber=dno

Figure 4.1: Feedback produced by SQL-Tutor for problem 1

When the student enters his/her incorrect solution, the messages generated by SQL-Tutor may not be helpful, especially if the student does not understand certain concepts. In this example, if the student does not fully understand the concept of join conditions, the messages may not be of much use. In this situation, a human tutor would present to the student an example and explain each step of the example. Figure 4.2 is a log of goals, actions and results that the tutor would perform and would look for, when explaining the example to the student.

GOAL : Determine if student knows which tables are relevant to
for the query. If not, point out to them which tables are
and why
ACTION : Ask the student to point out the relevant tables
EXPECTED RESULT : Student learns how to select relevant tables

GOAL : Determine if student understands the join condition. If not
draw example and use it to illustrate the join condition
ACTION : Tutor sketches the relevant tables and uses it to illustrate
the join condition
EXPECTED RESULT : Student understands the join condition

GOAL : Determine if student knows how to select the correct attributes
ACTION : Display the relevant part of the question that relates to
the selection of the correct attributes
EXPECTED RESULT : Student understands how to select the correct
attributes

Figure 4.2: Log of goals, actions and expected results of a human tutor

The advantage of doing it in steps is so that it is easier for the student to understand and to allow the student to acquire the mental process of planning (learning how to decompose problems into subgoals) and implementing it. By viewing each step of the example, the student is able to determine which step he/she might have gone wrong. By drawing the relevant tables and attributes, students are able to visualize the relevant tables and attributes. There are also other advantages to drawing the relevant tables and attributes, such as, people generally prefer pictures, there are no language barriers and it eases the memory load. This method of using examples could potentially help the student learn better and will be implemented in SQL-Tutor.

This method of using examples could be implemented in SQL-Tutor. The process in which the tutor explains the example can be shown as a sequence of slides. Figure 4.3, Figure 4.4, Figure 4.5 and Figure 4.6 shows the same process as described in the Figure 4.2. On the top of the Figures 4.3 to Figure 4.6, a subgoal is displayed, followed by the keywords to look for highlighted in red, and finally the sketch of the relevant tables. Figure 4.3 shows the tables in the database and highlights the relevant tables. Figure 4.4 identifies the keywords that would suggest to the student what the relevant tables are. Figure 4.5 displays the joining condition between the tables. Finally, Figure 4.6 highlights in red the attributes that needs to be retrieved (last name and address).

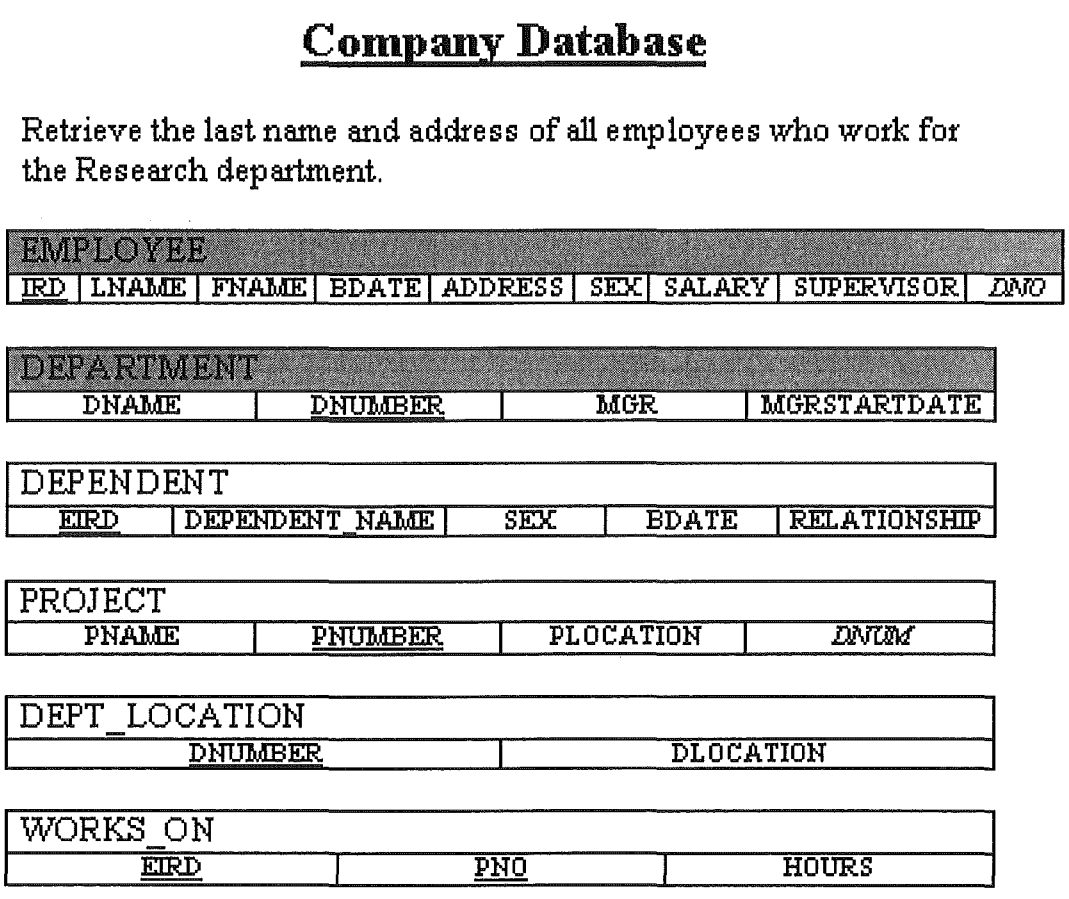

Figure 4.3: Slide 1 of the example

Examples for the first twenty questions for the company database was developed. However, for complex queries (queries that involve using the group by and having clauses), there are more attributes, tables and rela-

Locate relevant tables

Retrieve the last name and address of all **employees** who work for the Research **department**.

EMPLOYEE								
IRD	FNAME	LNAME	BDATE	ADDRESS	SEX	SALARY	SUPERVISOR	*DNO*
123456789	John	Doe	1/1/1999	2 F Street	M	40000	234567890	1
345678901	John	Smith	2/3/1978	3 G Street	M	20000	555555555	3
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
123467890	Jane	Borg	2/4/1978	3 A Road	F	20000	456789012	4

DEPARTMENT			
DNAME	DNUMBER	MGR	MGRSTARTDATE
Research	3	444443214	2/2/1978
...	...	...	...
Marketing	4	666666666	3/4/1993

Figure 4.4: Slide 2 of the example

Find employees with the same DNO as the Research department's DNUMBER

Retrieve the last name and address of all **employees who work for the Research department**.

EMPLOYEE

IRD	FNAME	LNAME	BDATE	ADDRESS	SEX	SALARY	SUPERVISOR	*DNO*
123456789	John	Doe	1/1/1999	2 F Street	M	40000	234567890	1
345678901	John	Smith	2/3/1978	3 G Street	M	20000	555555555	3
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
123467890	Jane	Borg	2/4/1978	3 A Road	F	20000	456789012	4

DEPARTMENT

DNAME	DNUMBER	MGR	MGRSTARTDATE
Research	3	444443214	2/2/1978
...	...	...	...
Marketing	4	666666666	3/4/1993

Figure 4.5: Slide 3 of the example

Display the names and addresses of relevant employees

Retrieve the **last name and address** of all employees who work for the Research department.

EMPLOYEE

IRD	FNAME	LNAME	BDATE	ADDRESS	SEX	SALARY	SUPERVISOR	*DNO*
123456789	John	Doe	1/1/1999	2 F Street	M	40000	234567890	1
345678901	John		2/3/1978		M	20000	555555555	3
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
123467890	Jane	Borg	2/4/1978	3 A Road	F	20000	456789012	4

DEPARTMENT

DNAME	DNUMBER	MGR	MGRSTARTDATE
Research	3	444443214	2/2/1978
...	...	...	...
Marketing	4	666666666	3/4/1993

Figure 4.6: Slide 4 of the example

tionships between tables are involved. A novice might find it difficult to comprehend the slides and in this situation, it would be ideal to have a human tutor present to tutor the student. It was decided that until better representations for examples are found, examples would not be implemented in SQL-Tutor.

4.1.1 Factors that affect feedback

The second finding was that tutors looked at four main factors when deciding what reply or feedback to give. These factors are: the proficiency of the student in the subject matter, the difficulty of the subject matter, the background of the student in the subject matter, and the level of motivation the student has. There are other factors such as facial expression, gestures, voice tone, etc..., that the tutor looks for, but is not considered as important as the main four factors. The list below explains in greater detail the four main factors.

- Proficiency : The proficiency of the student in the subject matter measures the overall level of understanding that the student has in subject matter (SQL). If the student has a low level of understanding in the subject matter, tutors tend to provide more complete feedback (perhaps the partial solution instead of a hint), and vice-versa. Human tutors usually determine the proficiency of the student based on previous encounters, measuring it by how well the student knows the subject. In SQL-Tutor, the proficiency of the student can be estimated by the number of problems solved over the number of attempts taken to solve the problems.

- Problem difficulty : The level of difficulty measures how difficult the problem is. Human tutors find that as the problem difficulty increases, they tend to spend more time and efforts into explaining the solution and the steps needed to derive the solution. This would suggest that tutor tend to provide vague feedback for easier problems, and feedback with more information content for difficult problems. If the tutor decides that the problem is too difficult, the student will be recommended to try another problem. Human tutors classify problems by an ad-hoc method. In SQL-Tutor, every problem is already assigned a difficulty level, ranging from one to ten, with problems of difficulty level one being the easiest problem.

- Background : The background of the student in the subject matter refers to the prior experience the student has in the subject matter. Human tutors usually determine the background of the student through informal contact. If the tutor thinks that the student has a good background in the subject matter, the tutor would provide less

hints compared to the number of hints a student who does not have a good background. Trying to measure the background of a student may prove to be difficult for SQL-Tutor. As mentioned above, tutors determine the background of the student mainly through informal contact. Currently, SQL-Tutor is not able to do this. An alternative method to gauge the background of the student is to request the student enter what he or she thinks their background level is. There are advantages and disadvantages to using this method. The advantage is that students usually have a better knowledge of their background than tutors, and this allows a more accurate way of identifying the background level. The disadvantage of this method is that it is subjective. Students may incorrectly assume their background and this could affect the type of feedback they receive. Currently, SQL-Tutor requires students to state (selecting one from the following three options: Novice, Familiar or Experienced) their background level before interacting with the system. The background level is given a numeric value ranging from one to three, with novice having the value of one, familiar having a value of two and experienced having a value of three.

- Motivational level : Motivation is mentioned as one of the learning functions which stimulate learning [11]. The motivational level of a student depends on many factors such as confidence, curiosity, control, attention, satisfaction and relevance (of the topic to be taught) [12]. Human tutors are able to detect the motivational state of the student and takes that into consideration when tailoring feedback. A student with a high motivational level would receive a vague feedback that will motivate them to think. The detection of a student's motivational state is constrained by interface limitations (unable to accurately detect confidence, attention and so on). The simplest method to determine the motivational level of a student is by the number of problems the student attempts to solve.

The four factors mentioned above are not all of equal importance. The interviewed tutors view the proficiency and problem difficulty of equal importance. The background and motivational level of the student is of equal importance, but is of secondary importance when compared to the proficiency and problem difficulty. The following list orders the factors starting from the most important.

1. Proficiency in subject matter and Problem difficulty
2. Background and motivational level of student

By combining these four factors into a single variable (called ability) [14], tutors then split students into categories. The ability of a student is defined as:

$$ability = (proficiency * 5) + problemdifficulty + background + motivation \quad (4.1)$$

The proficiency variable from (4.1) is multiplied by five to normalize it, so that the proficiency variable has a range from zero to ten (same as the problem difficulty). Equation (4.1) is derived via an ad-hoc attempt. It may be that this equation does not match how human tutors determine the ability of the student, and should be viewed as a first step towards developing the correct equation (if there is one such equation) that human tutors use to determine the ability of students.

Students of roughly the same ability are grouped in the same category. Each category will receive feedback suitable for that particular category. Using the model developed by Okazaki [13] as a guide, to classify the state of a student, a model to map the student's level of ability with the appropriate feedback is developed. There are two main criteria for this model. Firstly, the students with high ability would receive less support through the information content of the feedback messages, while students with a low ability would receive more support. An example of this would be that a student with high ability would not be able to obtain the complete solution (unless he or she explicitly requests it), whereas a student with a low ability would be able to obtain the complete solution. The other criteria is that sequence of feedback messages should be given in increasing amounts of information. For example, a student will receive a hint before receiving the complete solution. This ensure that the student is not overwhelmed with information. The current feedback selection method in SQL-Tutor is shown in Figure 2.4. Figure 4.7 illustrates the proposed feedback selection method.

Once the ability of the student is worked out, one can infer the state the student is in and provide appropriate feedback. For example, if a student has an ability of six (state 1), he or she would receive feedback messages in the following order: Error flag, Hint, Example, Partial solution and Complete solution. The other feedback messages (Positive/Negative feedback, complete solution) are available if the student requests it.

This model of determining what feedback to provide based on the ability of the student is implemented in SQL-Tutor.

4.1.2 Determining when help is needed

Tutors do sometimes initiate discussions with student when they feel that the students require assistance. How tutor know when to offer help to the student is mostly based on physical or verbal cues such as facial expression, gestures, voice tone and other cues. An example of a cue would be a student

First Try	Error Flag	Error Flag	Error Flag	Positive/Negative	Positive/Negative
Second Try	Hint	Hint	Hint	Error Flag	Error Flag
Third Try	Example	Example	Example	Hint	Hint
Fourth Try	Partial Solution	Partial Solution	Partial Solution	Example	
Fifth Try	Complete Solution				
	State 1	State 2	State 3	State 4	State 5

Figure 4.7: Proposed feedback selection method in SQL-Tutor

staring blankly at a screen. By initiating a discussion, tutors hope to correct any misconceptions the student might have, setting them on the correct solution path.

Due to the limitations in the communication channel between the student and the ITS (the only feedback the ITS can receive from the student is what the student types in), it would be difficult to observe the necessary cues. A crude way of trying to estimate when to provide help is by timing how long a student takes to complete a question. If a student takes longer than average amount of time, a message will appear, prompting the student if he or she requires assistance. This is a crude way of estimating if the student requires help and is prone to problems. An example of such a problem is if a student does not interact with the system for the average amount of time, the system thinks that the student requires assistance and provides it when it is not needed. Another problem with this approach is the problem of trying to determine the average amount of time needed to solve a problem. It should take into account the level of difficulty for the particular problem and adjust the average time appropriately.

Until a more accurate method of determining when students require help is discovered, this feature will not be implemented in SQL-Tutor.

4.1.3 Population learning

When a group of students encounter the same difficulty with a problem, this usually indicates that students share a common misconception or several misconceptions. In this situation, a human tutor would give a general rundown of the question to the entire class. This general rundown of the question would provide the entire class to be aware of the possible misconceptions that they may have. A similar method could be implemented in SQL-Tutor, where if a certain number of students violate a same set of constraints, all students using the system will be informed of the common error. In SQL-Tutor, each student has an individual student model that keeps track of what constraints the student has mastered or violated. By examining each student model, one could determine if a certain set of constraints get violated, and provide appropriate feedback to the entire population. There are certain problems to this approach such as: How many students should encounter the same difficulty before providing appropriate feedback? The population of students using SQL-Tutor is not fixed and can vary, so no specific number of students can be chosen. Due to the problems mentioned above, this feature will not be implemented in SQL-Tutor.

Chapter 5

Evaluation Study

After implementing the changes, it is crucial to determine if they contribute to student's learning. The aim of this evaluation is to access the contribution the new feedback selection method made, in particular:

- Did the new feedback selection method select appropriate feedback?
- Would the student prefer to select their own feedback or have feedback automatically selected by the system?

The rest of this section describes the method, which constitutes the design and physical makeup of the experiment. That is followed by the results and finally the discussion which interprets the results within the context of the two questions asked.

5.1 Method

5.1.1 Subjects

The subjects were students who were enrolled in the second year Introduction to Database (COSC226) course. All students had several lectures on SQL before they were able to interact with SQL-Tutor. Some subjects may have previous exposure to SQL. The subjects were randomly assigned to either the control or experimental group. The subjects in the control group interacted with the current version of SQL-Tutor, while the subjects in the experimental group interacted with the new version of SQL-Tutor. There were a total of 55 subjects in both the control and experimental group[1]. The experiment was conducted in the computer laboratories of the computer science department.

[1]There were more than 55 subjects in the experiment. However, the data collected for several subjects had to be discarded due to inconsistencies.

5.1.2 Design and Procedure

The evaluation consists of several data collection sessions, the pre-test, system interaction and post-test. Figure 5.1 illustrates the data collection sessions. Before students are able to interact with the system, they had to complete a pre-test. They were then allowed to interact with the system. When the evaluation period ended, the students were asked to complete a post-test. The aim of the pre-test and the post-test is to determine the subject's knowledge of SQL and also to find out if there are any significant differences between the control and experimental group.

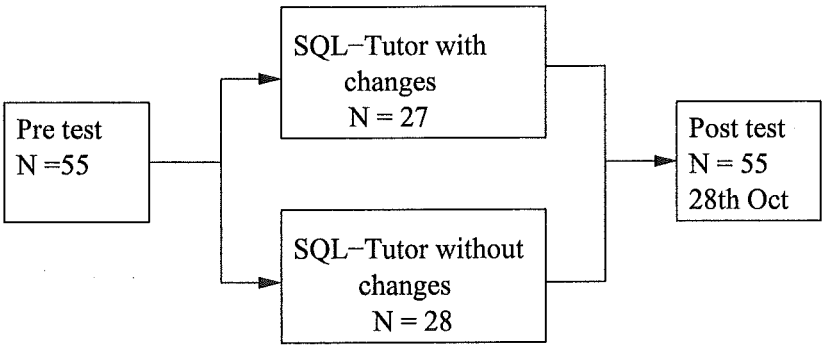

Figure 5.1: Experiment design

The pre-test and post-test (included in the Appendix B) each consists of three multiple choice questions. These questions were designed to evaluate the subject's knowledge of SQL. The marks allocated for the first, second and third questions were 1, 5 and 1 marks respectively. After answering each question, subjects recorded how confident they were of their answers using a three point Likert scale. The pre-test was given to the subjects when they logged on to the system for the first time.

After the subjects had submitted the pre-test, they were allowed to interact with the respective version of SQL-Tutor. The subjects were able to choose the problems they wish to solve, when they wish to log out and so on. The idea behind this is to have data collection that takes place not in an artificially created environment where subjects are constrained to act a certain way, but rather in a normal enviroment, where subjects have complete freedom. Every action that the subjects made was recorded automatically by the system in the form of student logs. The time limit in which the data collection took place occurred over thirty days. Since the subjects were able to choose when they wish to interact with the system (how many problems they wished to solve), having a data collecting session starting from the 4^{th} of September to the 28^{th} of October. The long data collection period ensured that the student logs were of sufficient length to perform any analysis

on.

On the 28^{th} of October, subjects took the post-test. The post test was included as part of the final exam for the course.

5.2 Results

The results were obtained from three separate sources: the pre-test, student logs and the post-test. The results from each source is discussed.

5.2.1 Pre-test

A total of 55 subjects participated in the pre-test. 28 subjects belonged to the control group, while the rest (27) belonged to the experimental group. The objective of the pre-test is to determine the knowledge level of the subjects. The ideal situation is that there is no significant difference in the knowledge level between both groups. As mentioned in Section 5.1.2, the pre-test consists of three questions, worth 1, 5 and 1 mark respectively. Figure 5.2 shows the mean scores for the pre-test. On average, the subjects belonging to the experimental group did slightly better than the subjects in the control group for questions one and three. The mean score for the control and experimental group was 3.97 and 4.13 respectively.

		Control Group	Experimental Group
Scores			
	Q1	0.39	0.44
	Q2	2.92	2.91
	Q3	0.66	0.78
	Total	3.97	4.13
Confidence level			
	Q1	1.35	1.44
	Q2	0.8	1.16
	Q3	1.14	1.36
	Total	3.29	3.96

Figure 5.2: Mean scores for the pre-test

Subjects in both groups were also asked how confident they were of their answers to the questions in the pre-test. Subjects are able to choose their confidence level from a scale of 0 (not confident) to 2 (very confident). Results indicate that subjects in the control group were slightly less confident than subjects in the experimental group. The mean confidence level of subjects in the control group was 3.29 and the mean confidence level of subjects in the control group was 3.96

To determine if there is a statistical difference in the knowledge level between the two groups, a two tailed t test was carried out. The null hypotheses for the t test would be that there is no difference between the means at the 0.05 level of significance. The alternative hypotheses is that there is a difference between the means at the 0.05 level of significance. The results indicate that there is no significant difference between the mean scores of the two groups ($t = 0.35$, $p > 0.05$) and hence the null hypotheses is accepted.

5.2.2 Student logs

Every action performed by a subject during the evaluation period was recorded in a log, along with a time stamp. Figure 5.3 shows a portion of a log file for a subject. It shows a subject logging on to SQL-Tutor for the first time. Problem 59 was the first problem the subject attempted. The subject solved the problem correctly on the first attempt.

Once analyzed, the log files can be used to answer the questions posed in Section 5. Figure 5.4 presents a summary of the variables that were analyzed.

The average amount of time the subjects in the control group spent interacting with the system was 55.89 minutes. However, it is difficult to determine if subject actually spend all the time attempting to solve the problem or if they are doing something else. An example of this situation is if a student attempts a question, but leaves for a short break instead of spending the time attempting to solve the question. Subjects in the experimental group spent an average of 83.9 minutes interacting with the system. This indicates that subjects in the experimental group were more willing to spend time interacting with the system. Another possible reason why subjects in the experimental group spent more time interacting with the system was that the feedback provided was not what they expected and wasted time selecting the type of feedback they are expecting.

This issue can be clarified up by examining the number of problems attempted by the subjects, the number of solved problems and the attempts to solve problems that could not be solved on the first attempt. If the subjects in the experimental group attempted and solved more problems then the subjects in the control group, we would know that the system providing inappropriate feedback does not really affect the amount of time the subject spent interacting with the system. By examining the attempts to solve problems that could be solved on the first attempt, we are able to determine the usefulness of the feedback. If subjects in the experimental group requires more attempts to solve a problem, it suggests that the feedback provided to this group is not particularly useful.

On average, subjects in the control group attempted and solved less problems than the subjects in the experimental group. Subjects in the control group attempted an average of 10.5 problems and managed to solve an average of 7.35 problems, while the subjects in the experimental group attempted an average of 15.5 problems and managed to solve an average of 11.73 problems. This shows that subjects in the experimental group attempted and learnt more about SQL than the control group. However, this conclusion may not be true in certain cases. For example, if subjects receives the complete solution or partial solution as feedback, they may not attempt to solve the problem, even though they do know the correct solution. Taking this into account, the average number of solved problems for the control group increased to 8 and 13 for the experimental group. Using this new measure of solved problems, the percentage of solved problems over attempted problems for the control and experimental groups are 0.76 and 0.85 respectively.

In order to evaluate how useful the feedback selected by the system is, we should examine the average number of attempts taken to solve problems that could not be solved on the first attempt. This figure is a good measure of the usefulness of the feedback. For example, if a subject requires many attempts before the problem is solved, this would indicate that the feedback provided to the subject is not very useful.

```
-----------
13:52:48 2/10/2000
Log in
-----------
13:53:20 2/10/2000
first-time-user

-----------
13:53:20 2/10/2000
Mode set to 4

-----------
14:00:28 2/10/2000
Changing database to registration
 Problem number is 59

-----------
14:00:28 2/10/2000
drawing: problem is 59 its status is NEW
-----------
14:01:25 2/10/2000
Pre-process:

Help Level 0
Feedback Option: Feedback
Database: registration
Problem number: 59
Their attempt:
Select: *
From: vehicle
Where:
Group by:
Having:
Order by:
Two-level-help?: NIL
Mode: 4

tries: 1
proficiency: 0.0
motivation: 1
ability: 0.0
-----------
14:01:25 2/10/2000
set ability to 12.0

14:01:25 2/10/2000
Answer correct
```

Figure 5.3: Part of a student log

	Mean		Standard Deviation	
	Control Group	Experimental Group	Control Group	Experimental Group
Total Interaction Time (mins)	55.89	83.9	68.36	80.59
Number of attempts	10.5	15.5	9.8	13.96
Number of solved problems	7.35	11.73	7.52	12.29
Attempts to solve problems that could not solved on the first attempt	1.72	3.77	1.04	3.58

Figure 5.4: Mean interaction details

The average number of attempts taken to solve problems that could not be solved on the first attempt was 1.72 and 3.77 for the control and experimental group respectively. This result indicates that subjects in the experimental group persisted in trying to solve problems that initially could not be solved. Another interpretation of this result could be that the feedback provided by the system in the control group is more useful than feedback provided by the system in the experimental group, hence, subjects in the control group are able to solve the problem in less attempts. However, it could just be that students in the control group requests the complete solution whereas subjects in the experimental group may just wait accept feedback provided by the system, instead of requesting specific feedback. The student logs will need to be examined in greater detail to determine if this is so.

After the student logs were examined, it was found that subjects in the control group requested specific feedback less than subjects in the experimental group. This suggests that the feedback provided to the subjects in the experimental group was not as useful as the feedback provided to the subjects in the control group. To determine if the the changes contributed to the subjects' learning, a post test will be conducted to determine the knowledge the subjects have in SQL.

5.2.3 Post-test

The main objective and format of the post-test is identical to the pre-test: to determine the knowledge the subject has in SQL. We expect to find that there is a significant difference between the knowledge level of subjects from the control and experimental group, with subjects from the experimental group having a better knowledge of SQL. All 55 subjects who participated in the pre-test, completed the post-test. Figure 5.5 shows the mean scores for the post-test. Subjects in the control group scored an average of 4.64 out of a possible 7, while subjects in the control group scored an average of 5.07. A two tailed t test was carried out to determine if there is a statistical difference between the two groups. The results of the t test indicates that there is no significant difference between the mean scores of the two groups ($t = 1.387$, $p > 0.05$).

5.3 Discussion

The pre-test showed that there are no significant differences in the average knowledge levels of subjects in both the control and experimental groups. Examination of the students' log revealed that subjects in the experimental group attempted and completed more problems. However, the feedback to the experimental group seemed less useful than the feedback provided to the control group. This indicates that the sequences of feedback proposed for

	Control Group	Experimental Group
Scores		
Q1	0.82	0.77
Q2	3.28	3.4
Q3	0.53	0.88
Total	4.64	5.07

Figure 5.5: Mean scores for the post-test

students of different categories of ability were not the ideal sequences. The post-test revealed that after interacting with the system, both groups had a better knowledge of SQL. However, there was no significant difference in the knowledge level between the two groups.

It is evident that the selection of feedback for the experimental group failed to provide the expected individualization. Despite the lack of encouraging results, it should be noted that this is a rough model of how human tutors individualize feedback and is possible that with a more sophisticated model of how human tutors adapt feedback, the gains of individualization will be realised.

Chapter 6

Conclusion

Since that late 1950s, computers have been used to tutor students in various domains. These early tutoring systems were not able to effectively adapt instructions to suit individual students and this led the development of ITSs. The main goal of ITSs is to provide individualized feedback. However, most ITSs are not able to do this as well as a human tutor.

A series of interviews with tutors took place to determine what factors human tutors take into account when deciding what sort of feedback to provide. Tutors took note of how proficient the student is in the subject matter, the problem difficulty, the background of the student in subject matter and the motivation of the student, when deciding on what sort of feedback to provide.

A model of how human tutor decide what sort of feedback to provide is constructed and implemented into SQL-Tutor. An evaluation study of the changes was conducted and it revealed that the changes made did not significantly contribute to the subject's learning. If a more sophisticated model could be developed, taking into account more factors, it may possibly have a significant effect on the student's learning experience.

Acknowledgements

I would like to thank Dr Antonija Mitrovic for guiding and assisting with this research, Kurt J Hausler for implementing the changes to SQL-Tutor, Jane McKenzie for proof reading this report, and to the tutors who agreed to be interviewed.

Bibliography

[1] Bloom, B. S. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher, 13, pp. 3-16. 1948.

[2] Burns, H. L. and Capps, C. G. Foundations of Intelligent Tutoring Systems: An Introduction. Foundations of Intelligent Tutoring Systems. Lawrence Erlbaum Associates, Hillsdale, NJ. 1988.

[3] Woolf, B. AI in Education. Encyclopedia of Artificial Intelligence, Shapiro, S., ed., John Wiley & Sons, Inc., New York, pp. 434-444. 1992.

[4] Elmasri, R. and Navathe, S. B., Fundamentals of database systems (2nd ed) Benjamin / Cummings, Redwood, CA. 1994.

[5] Mitrovic, A., SQL-Tutor: a Preliminary Report, Technical Report TR-COSC 08/97, Computer Science Department, University of Canterbury, 1997.

[6] Mitrovic, A., Learning SQL with a computerised tutor, Proc. 29th SIGCSE Tech. 307-311. 1998.

[7] Gertner, A, Conati, C, and VanLehn, K., Procedural help in Andes: Generating hints using a Bayesian network student model. Proc. 15th National Conference on Artificial Intelligence. Madison, Wisconsin. 1998.

[8] Pamela J. Woods and James R. Warren., Adapting Teaching Strategies in Intelligent Tutoring Systems. ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal. 1996.

[9] Hume, Gregory D., Using Student Modelling to Determine When and How to Hint in an Intelligent Tutoring System, Ph.D. , Illinois Institute of Technology. 1995.

[10] Joseph E. Beck and Beverly P. Woolf, Using a Learning Agent with a Student Model. ITS'98, San Antonio, USA. 1998.

[11] Shuell, T, J., Designing instructional computing systems for meaningful learning., Foundations and Frontiers in Instructional Computing Systems (in press).

[12] Teresa, D, Soldato., Detecting and reacting to the learner's motivational state, ITS 92, Montreal, Canada, 1992.

[13] Okazaki, Y., Watanabe, K. and Kondo, H., An Implementation of the WWW Based ITS for Guiding Differential Calculations. Proc of workshop: Intelligent Educational Systems on the World Wide Web". Proc. 8th World Conference of the AIED Society, Kobe, Japan. 18-22. 1997.

[14] Akira, Takeuchi. and Setsuko, Otsuki., EXPITS: an Experimental Environment on ITS, ITS 92, Montreal, Canada, 1992.

Appendix A

Four tutors, selected on the basis that they had at least a few years of tutoring, were interviewed. A brief overview of the project was given to the tutors. The table below summarises the questions, reasons why the question was asked, answers and analysis of the answers. Note that the tutor responses have been combined and edited where appropriate.

QUESTION	REASON	ANSWER	ROUGH ANALYSIS
Roughly how long have you been tutoring ?	To gauge how much weight should be placed on their answers.	All the tutors had several years of experience tutoring database material.	Their answers carry roughly the same weight.
Do you find that students have problems with the concepts (what it means) rather than how to go about doing the question ?	Students are assumed to know the background knowledge and concepts, but is this assumption actually true ?	Students are expected to know the concepts. They usually have problems with concepts rather than how to go about doing the question.	Students can roughly be divided into two groups. The first group of students include students who have learnt and remember the background materials and key concepts. This means that unnecessary material should not be presented to this group. The other group includes students who have never really learnt the background material and concepts, of have forgotten them. This group should be presented with basic information.

QUESTION	REASON	ANSWER	ROUGH ANALYSIS
Do you initiate any discussions with students ?	To find if tutors do initiate discussions with students and why.	Sometimes, depending on whether the class is big or not.	In ITS, the size of the class does not matter. Human tutors are able to determine if a student is lagging behind and approach the student. ITS suffers from the limitations in bandwidth of the communication channel and the only feedback that it receives is what the student types in. One way of trying to measure if a student is behind is to time how long he or she takes to complete a question. If the student takes longer than the average time (obtained by taking the average time that all students take to complete that question), perhaps SQL-Tutor could provide a "pop-up" message asking the student is he or she needs help. This is a crude way of estimating the pace of the students and this method can be easily "fooled".

QUESTION	REASON	ANSWER	ROUGH ANALYSIS
When students request assistance, what are the typical statements they say ?	To find if students know what they want help on.	They usually say something like "Here is what I have done so far. Am I doing it right?" or "I do not know how to do this problem. Could you help me?"	SQL-Tutor would have to look at where students have gone wrong and inform the student.
From the statement the student makes, do you look for certain key words that help you determine a particular way to help a student? (e.g. If you hear the word EXPLAIN, do you tend to explain to the student the soultion rather than giving them a hint?)	To find out if student do state which method they prefer their feedback in.	Yes. For example, if they mention that they would like an explanation, the tutor would switch from whatever method of feedback they planned and give an explanation.	SQL-Tutor should have a range of feedback methods avaliable and students wanting a particular method of feedback can choose it. SQL-Tutor should keep a record of which feedback method a student prefers, so that when they next log on, there would be no need to ask them which feedback method they prefer. A flexible way for students to change their preferred method should be provided.
If the student does not clearly state his or her intention (i.e. does not state if he or she wishes to know if they are on the right track, or where they might have gone wrong), how do you go about trying to understand what the student wishes to gain out of the discussion?	To find out what a student wants if he or she does not mention it.	Ask them questions to clarify what they want.	SQL-Tutor currently has five levels of feedback, which the student can select. If a student does not select a feedback level, SQL-Tutor automatically selects the first level of feedback (positive or negative), then the second level of feedback (error flag) and finally stopping at the third level of feedback (hint). SQL-Tutor could be modified so that it does not automatically select the level of feedback, but decides which feedback level to use based on the student model.

QUESTION	REASON	ANSWER	ROUGH ANALYSIS
When you approach a student, what factors do you take into account when deciding how to reply? 1. Difficulty level of the problem. 2. Motivation of student. 3. Pace of student. 4. How much time you think the problem deserves. 5. The student's level of proficiency with the topic. 6. Other factors.	To find out what factors human tutors take into account when deciding on how to reply.	All of the below: 1. If the problem has a high level of difficulty, more time would be spent. If the problem is a simple problem, leading questions would be asked, hoping to lead the student to the correct solution. 2. Yes. If the student is motivated and wishes to learn more, more time would be spent. The amount of time spent with a student also depends on how large the class is. 3. Yes. If a student is behind, fewer leading questions will be asked and straightforward explanations will be given. 4. When dealing with large classes, tutors tend to move between students quickly. 5. Tend to give students feedback that they are able to use. (i.e. matches their level of proficiency on the topic).	The student model in SQL-Tutor could be extended to take note of the level of difficulty of the question, the motivation and pace of a student and the level of proficiency of the student.

QUESTION	REASON	ANSWER	ROUGH ANALYSIS
Do you form some sort of plan of how each student is faring ?	To find if human tutors do form and keep a plan of how a student is faring.	Yes.	SQL-Tutor has a student model which is the equivalent of a human tutor's mental 'plan'. However, the student model in SQL-Tutor is not as complicated as a human tutor's plan and should be extended to consider factors such as how well a student retains information over time, how long a student takes to master a concept, etc. The student model also has to take into account how difficult the question is, how motivated the student is, how well the student is faring and so on.
How do you determine if a student has understood what you have said to him or her?	Ideally, tutors should stop explaining once the student has understood the concept. This question is to find out how human tutors know when to stop explaining.	Get the student to solve a similar problem and if he or she is able to do it, that means that they understood what you have said to them. Another method is to ask them some questions about what you have just told them.	SQL-Tutor could determine if a student has understood a concept by asking the student to solve a similar question.
What if a student does not understand what you have said to him or her? Do you 1. Explain in more detail. 2. Use another method. 3. Brush off and recommend another problem. 4. Try another method.	To find how a tutor should reply when a student does not understand the reply that the tutor gave.	If the student does not understand what was said, another method would be used. If that does not work, the answer would be shown to the student, the example closed, and the student asked to work on a similar problem.	SQL-Tutor should have several feedback methods, so that if a student does not understand the reply, SQL-Tutor can fall back on another feedback method. Also, students tend to respond better if they see a visual representation (as opposed to just a verbal representation) of how the query should be done and how it is processed.

QUESTION	REASON	ANSWER	ROUGH ANALYSIS
If a few students have encountered difficulty with the same problem, what would you do? Do you spend extra time explaining the problem?	When a few students have difficulty with a question, this usually indicates that something might have gone wrong somewhere and this question is to find out how human tutors deal with this.	If a few students encountered the same problem with a question, a general rundown of the question would be given to the entire class. Extra time would be spent if students still have difficulty with the problem.	If more that a few students have difficulties with a question, SQL-Tutor should recognise this and perhaps give a message to other students attempting the same question, informing them that other students have difficulties with this question and
Do you find that students have difficulty with more complex queries (queries that need at least two sub-skills to solve, i.e. a query that required the use of the *group-by* statement) ?	To confirm that students have more difficulty with more complex queries.	Yes. Students tend to have more difficulty with complex queries.	This suggest that more attention should be paid to the feedback methods and levels of more complex queries.
A question usually has a concept or point to get across to the student. For complex queries, one has to know certain sub-concepts before attempting the question. Suppose a student is attempting to do a complex query and has made two mistakes: one mistake with the concept that the question is trying to get across and the other mistake is with a sub-skill, how do you go about helping the student with the problem ? For this question, a problem dealing with a complex query and a incorrect answer was shown to the tutors. They were asked how they would go about helping a student with that answer.	To find out how human tutors deal with multiple errors.	If a student has problems with a complex query, his or her previous work on that question would be examined. The basic "building blocks" would be examined first to determine that the student has all the sub-skills to solve the question. After that, the explanation will be built on those basic sub-skills and get progressively closer to the concept that the question was trying to get across (i.e. for a query that needed to use the group by clause of the select statement, they first explained how the basic clauses (select, where, from), then go on to the more complex clauses). Trying to visually show how the query was done is also useful.	Human tutors build up the basic "building blocks" and uses these "blocks" as a foundation to work on more complex queries. SQL-Tutor should follow this method when dealing with complex queries.

Appendix B

SQLT-Web
SQL-Tutor on the Web

Introductory Questionnaire

SQLT-Web is an *intelligent tutoring system* that supports interactive learningvia WWW. The system adapts to the style, learning abilities and needs of people using it. In order to be able to achieve that, we ask you to answer a few questions.

The questions are based on a small database, containing only one table, MOVIE. The MOVIE table contains information about movies held in a video club. Each movie has a unique number, which is the primary key of the table. Additionally, we have the title (TITLE), the year of production (YEAR), the critic's rating (CRITICS), the type of movie (TYPE), the number of Academy awards the movie was nominated for (AANOM) and has won (AAWON), and the number allocated to the director of the movie (DIRECTOR).

MOVIE (NUMBER, TITLE, YEAR, CRITICS, TYPE, AANOM, AAWON, DIRECTOR)

Please answer ALL questions!

1. We need to find the titles of all movies other than comedies. The following SQL statement has been suggested as an answer:

SELECT TITLE
FROM MOVIE
WHERE TYPE = NOT('comedy')

Is this statement correct?

Yes
No

How confident are you of your answer?

Very confident
Moderately confident
Not confident

---

2. Now, we need to find the title of the movie which has won the most awards. Select ALL correct answers:

SELECT TITLE
FROM MOVIE
WHERE AAWON = MAX(AAWON)

SELECT TITLE
FROM MOVIE
GROUP BY NUMBER
HAVING AAWON = MAX(AAWON)

SELECT TITLE
FROM MOVIE
WHERE AAWON = (SELECT MAX(AAWON) FROM MOVIE)

SELECT TITLE
FROM MOVIE
GROUP BY TITLE
HAVING AAWON = (SELECT MAX(AAWON) FROM MOVIE)

SELECT TITLE

```
FROM MOVIE
WHERE AAWON >= ALL (SELECT AAWON FROM MOVIE)
```

How confident are you of your answer?

Very confident
Moderately confident
Not confident

---

3. We need to find the total number of awards won by comedies in 1983. Which of the following statements will achieve that?

```
SELECT SUM(AAWON)
FROM MOVIE
GROUP BY TYPE
HAVING TYPE IN ('comedy') AND YEAR=1983
```

```
SELECT SUM(AAWON)
FROM MOVIE
WHERE TYPE='comedy' AND YEAR=1983
```

```
SELECT SUM(AAWON)
FROM MOVIE
WHERE TYPE='comedy' AND YEAR=1983
GROUP BY NUMBER
```

How confident are you of your answer?

Very confident
Moderately confident
Not confident

Submit

Post-test

The questions are based on the MOVIE table. Each movie has a unique number, which is the primary key of the table. Additionally, we have the title, the year of production (YEAR), the critic's rating (CRITICS), the type of movie, the number of Academy awards the movie was nominated for (AANOM) and has won (AAWON), and the number allocated to the director of the movie (DIRECTOR). Please answer ALL questions:

1. We need to find the titles of all comedies or dramas. Is the following SQL statement correct? Yes/No

```
SELECT TITLE
FROM MOVIE
WHERE TYPE='comedy' OR 'drama'
```

2. What is the type of movie that had the highest number of movies made in 1980? Select ALL correct answers:

a.
```
SELECT TYPE
FROM MOVIE
WHERE YEAR=1980
GROUP BY TYPE
HAVING MAX(COUNT(*))
```

b.
```
SELECT TYPE
FROM MOVIE
WHERE YEAR=1980
GROUP BY TYPE
HAVING COUNT(*)>=ALL (SELECT COUNT(*) FROM MOVIE WHERE YEAR=1980 GROUP BY TYPE)
```

c.
```
SELECT TYPE
FROM MOVIE
WHERE YEAR=1980 AND COUNT(*) = (SELECT MAX(COUNT(*)) FROM MOVIE WHERE YEAR=1980)
GROUP BY TYPE
```

d.
```
SELECT TYPE, MAX(COUNT(*))
FROM MOVIE
WHERE YEAR=1980
GROUP BY TYPE
```

a.
```
SELECT TYPE
FROM MOVIE
WHERE YEAR=1980 AND NUMBER = MAX (COUNT(*))
```

3. We need to find the total number of awards won by comedies in 1983. Select all of the following statements that will achieve that?

a)
```
SELECT COUNT(*)
FROM MOVIE
WHERE YEAR IN (1981, 1982, 1983) AND TYPE='drama'
```

b)
```
SELECT COUNT(*)
FROM MOVIE
WHERE TYPE='drama'
GROUP BY YEAR
HAVING YEAR=1983 OR YEAR=1982 OR YEAR=1981
```

c)
```
SELECT COUNT(*)
FROM MOVIE
WHERE YEAR>=1981 AND YEAR<=1983
```